Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

An Order-Based Approach to Mission-Oriented Autonomous Robot Control: Managing Complexity, Merging Multiple Plans, and Performance Analysis Given Partial Probabilistic Information

Keyong Li [*] **Raffaello D'Andrea** [**]

[*] *Cornell University, M&AE Department, Ithaca, NY 14853, USA (email: likeyong@ieee.org)*
[**] *Cornell University, M&AE Department, Ithaca, NY 14853, USA (email: rd28@cornell.edu)*

Abstract: This paper formulates a framework for mission-oriented robot control. The benefit of this approach is multifacet. First, it suggests a promising direction for managing the complexity of encoding control policies. Second, it establishes a natural connection between feedback control and planning. Third, it allows the analysis of the time-to-mission-accomplishment distribution with almost minimum probabilistic information. This framework has been applied to the domain of robot soccer as a test of effectiveness. The resulting system competed at par in a simplified game setting against the strategy of a former championship team in the International RoboCup Competition.

1. INTRODUCTION

The past decade has seen some encouraging progress on applied autonomous robotics. However, as the problem domain expands, it becomes more and more difficult to achieve richer and more reliable behaviors without a wider recognition of and more practical solutions to the complexity issue associated with such systems. Although the artificial intelligence community has produced very impressive results on planning and learning, those techniques typically run into complexity problems when extended to the domain of autonomous robot control. This is in large part due to the continuous and uncertain nature of the work space of autonomous robots.

In this paper, we formulate a framework for mission-oriented robot control that is motivated by the consideration of complexity issues. We will call the control policies within this framework *order-based policies*, where the "order" is related to the concept of "priority". We do not use the widely used term "priority-based action selection" here because there are many differences. The policies discussed in this paper are hierarchical. But different from the conventional hierarchical approach, they have the novel feature that the higher level (to be made precise in this paper) does not directly reference the sensory inputs or even the state. Instead, the higher-level decision is formulated as nothing more than an ordering of the lower-level actions. The immediate consequence is that the number of policy candidates becomes determined by the number of lower-level modules. In other words, the richness of the lower-level becomes the main tuning-knob of the complexity/richness of the overall control policy. This tuning-knob helps us concentrate on a small pool of meaningful policy candidates on one hand, while allows all feasible policies to be included on the other. Moreover, it allows complexity to be adjusted at small increments. Thus, it appears to be an ideal tuning-knob for managing complexity.

We further show the advantage of the proposed framework from two additional aspects. First, we analyze the performance of the order-based policies in the presence of uncertainties. This turns out to be an interesting exercise, which led us to discover a criterion for comparing the convergence rates of stochastic processes with uncertain and time-varying transition probabilities. Second, we demonstrate the effectiveness of our approach by competing it against the legacy code of the former Cornell University RoboCup Team, who had won four Championships in the International RoboCup Competition, F180 (Small Size) League[2].

Many researchers have suggested interesting formulations of autonomous robot control. Some studied the linguistic aspect, see for example Andersson and Hristu-Varsakelis (2003); Brockett (1990); Egerstedt (2003); Manikonda et al. (1998). The elementary actions (EAs; to be formally introduced later) in the present paper play a role similar to that of the "atoms" in Andersson and Hristu-Varsakelis (2003). Both of them are building blocks of the overall control law. However, the atoms are designed to be executed in explicitly prescribed sequences, with each atom knowing when to terminate itself. This approach resembles people's common experience with programming languages. However, in the field of autonomous robot control, this approach cannot avoid complicated coupling among the designs of the atoms. In contrast, the EAs formulated in this paper know explicitly when they are ready to take charge, and they are terminated when a more preferable one becomes ready. These EAs are still arranged into strings, in which the order reflects priority of execution, but the actual sequence of their execution is implicit and depends on how the uncertainties play out.

Other important works on robot control architectures include Brooks (1991); de Sevin and Thalmann (2005); Isla et al. (2001); Lehman et al. (2006); Suh et al. (2004); Bryson (2002), just to name a few. First, consider the well-known subsumption architecture described in Brooks (1991). Aside from its underlying vision for the evolution of intelligent robot research and purely from an engineering standpoint, the subsumption architecture organizes behavior modules in a way such that all modules have access to the sensory inputs and make decisions in parallel, while

[1] Both authors gratefully acknowledge support from the Air Force under grant F49620-02-1-0388 and from the NSF under grant ECS-0329743.

[2] The code was adapted by one of its original designers for a 2-on-2 game. The actual competition was 5-on-5.

some relatively sophisticated modules may choose to "subsume" the decisions made by the more primitive modules. This parallel decision making and overriding mechanism is in a way related to the order-based approach that we propose here. However, there are many differences. Most importantly, in what we propose, which module gets to override another has nothing to do with their levels of sophistication, but is based on their individual goals. The arbitrator is what we call the "higher level", which can make its core decisions off-line (or at least at a much slower time scale than that of the lower level).The framework that we propose also shares a similar spirit with the priority-based architectures in Isla et al. (2001); Lehman et al. (2006); de Sevin and Thalmann (2005); Suh et al. (2004) in that they all formulate lower-level control modules to include some forms of preconditions. But the current paper went further to incorporate partial probabilistic information of the outcomes of the lower-level modules. Furthermore, the probabilistic information that we assume is almost minimum for enabling rigorous analysis.

2. COMPLEXITY OF AUTONOMOUS ROBOT CONTROL

The autonomous robot control system that we consider resides in a larger system depicted in Figure 1. Let's call this larger system *the world.* Let $x \in \mathcal{X}$ be the state

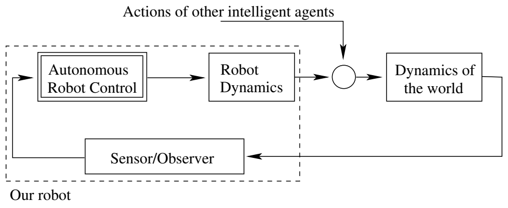

Fig. 1. The world of our robot.

vector of the world from the perspective of one robot, which typically includes the position and velocity of every relevant object as well as the states of the robot's own dynamics (e.g., the voltage applied to its motors). In practice, x is almost always digitized, and so we will assume. We also assume that x only contains components observable and relevant to this robot.

Let $x = (x^{(1)}, x^{(2)}, \ldots, x^{(N)})$, where each $x^{(i)}$ is a binary digit. The domain of x is thus $\mathcal{X} = \{0,1\}^N$. For control, let $u \in \mathcal{U}$ be the command issued by the robot to its actuators. We also assume that the command is a binary word $u = (u^{(1)}, u^{(2))}, \ldots, u^{(M)})$, hence $\mathcal{U} = \{0,1\}^M$. We will call x and u the *state word* and *command word*, respectively. We assume that the lengths of the state and command words are minimum for the precision needed. Without introducing hierarchies or any other method for managing complexity, the overall control law of the robot is a mapping

$$u = f(x) : \mathcal{X} \to \mathcal{U}. \qquad (1)$$

2.1 Notions of Complexity

Definition 1. Let $\mathcal{B} \subseteq \mathcal{U}^{\mathcal{X}}$ be a candidate set for the overall control policy. The *coding complexity* within $\mathcal{B}$, denoted by $\mathbb{CC}(\mathcal{B})$, is the minimum length of the binary block code that encodes the set $\mathcal{B}$. I.e., $\mathbb{CC}(\mathcal{B}) = \log_2 |\mathcal{B}|$, where $|\cdot|$ denotes the cardinality of the set.

Clearly, the coding complexity so defined is also the amount of information (in bits) needed for the process of selecting one control law within an a priori range of candidates.

The coding complexity needs to be distinguished from the *specification complexity* defined in Egerstedt (2003). Focusing on a framework where the control of a robot mission is a string over a *motion alphabet* (which is similar to the collection of elementary actions considered in later parts of this paper), specification complexity is defined as the length of the string multiplied by the base-2 logarithm of the cardinality of the motion alphabet. The difference between these two notions of complexity lies largely in the frameworks of the control laws on which we concentrate. Both frameworks provide ways of managing complexity. In Egerstedt (2003), this is done by managing the length of the control string that accomplishes the mission, whereas in this paper, complexity is managed by manipulating the collection of elementary actions.

Another relevant notion of complexity is the amount of computation needed for an algorithm to find the optimal behavior within $\mathcal{B}$, which depends on the design methodology and might be called *design complexity.* At most, it is on the order of $|\mathcal{B}|$, in which case the algorithm conducts an exhaustive search (e.g., a dynamic programming policy iteration). At least, it equals the coding complexity, which means the optimal behavior is available somewhere, and we simply need to copy it to the robot. This might be the case when the reference of the robot behavior is how a human expert would behave. In this paper, we mainly consider managing the coding complexity, while the design complexity is likely to be managed simultaneously.

2.2 Limitation of Complexity Reduction by Introducing Layers

This subsection motivate our study by analyzing the limitation of managing complexity by merely introducing layers.

For the control law in equation (1), the candidate set is $\mathcal{U}^{\mathcal{X}}$, whose cardinality is $(2^M)^{2^N}$. The coding complexity is thus $\mathbb{CC}(\mathcal{U}^{\mathcal{X}}) = M \cdot 2^N$. For problems that are of practical interest, M and N can easily be over one hundred, which makes the coding complexity extremely large. Moreover, N (the length of the state word) contributes to this complexity much more significantly than M (the length of the command word) does.

The primary approach for dealing with complexity has been formulating hierarchical structures. (We will use the words "hierarchy", "layer", and "level" interchangeably.) The idea of the conventional hierarchical approach is that the lower level controls will respond to the changes of certain state variables such that the dynamics associated with these variables can be neglected at the higher level. At the same time, the lower level would provide a set of control modules, which constitute a reformulated and likely more compact set of commands for the higher level to use.

Consider a two-layer case. Let the lengths of the state words at the higher level and the lower level be N_1 and N_2, and let there be L lower-level modules available in the command set of the higher level. Here, both N_1 and N_2 are less than N, although $N_1 + N_2$ is usually greater than N. The number L is usually much smaller than both 2^M and 2^N. The coding complexity of this two-layer system is $\mathbb{CC}(\text{2-layer policies}) = 2^{N_1} \log_2 L + L \cdot M \cdot 2^{N_2}$, which is often much lower than that of the original system since $N_1, N_2 < N$. Moreover, the second term might be replaced by a much smaller quantity. This is because the lower-level modules may be designed analytically, and the analysis may result in efficient parameterizations of their candidate sets. Suppose the lower-level modules are fixed a priori. Denote the candidate set of the overall control under this assumption by $\mathcal{B}_1$. The coding complexity of the system is then given by the first term only, i.e., $\mathbb{CC}(\mathcal{B}_1) = 2^{N_1}\log_2 L$. This can also be thought of as the coding complexity of the upper layer.

Although $\mathbb{CC}(\mathcal{B}_1)$ is certainly much less than $\mathbb{CC}(\mathcal{U}^{\mathcal{X}})$), is this complexity usually moderate enough? Under the conventional hierarchical approach, the state variables referenced by the higher-level typically include the velocities and positions of all robots and objects in the robot's world plus an array of internal states. Consider a seemingly simple case with 6 robots and 2 moving objects, assuming all positions and velocities are recorded with 8 bits for minimally acceptable precision. Then, $N_1 = 128$, and the corresponding coding complexity is on the order of 2^{128}, or 10^{39}. In other words, if the mission control policy is to be specified as a look-up table, then about 10^{39} binary entries need to be filled.

3. ORDER-BASED CONTROL POLICIES

In this section, we formulate a class of mission control policies which we call *order-based policies*. Our approach may be used to dramatically reduce the coding complexity of the policy. It also provides a way of fine-tuning the coding complexity after an initial policy design. What we propose involves a lower level and a higher level. The lower level produces elementary actions (EAs) together with their preconditions, postconditions and assessments of their minimum probabilities of "succeess". The higher level on the other hand produces a mission-accomplishing policy through ordering the EAs. To distinguish the lower-level controls and the overall control, we will call the lower-level ones "control laws", and call the overall control a "control policy".

3.1 Elementary Actions

Here, an EA is a 5-tuple $(f, \phi, \psi, \hat{p}, \Theta)$. Recalling that $x \in \mathcal{X}$ is the digitized state of the robot's *world*, the function $f(x) : \mathcal{X} \to \mathcal{U}$ is a lower-level control law that produces the command word directly fed to the actuators. Denote the state evolution under f by $F(x, w, \tau) : \mathcal{X} \times \Omega \times \mathbb{R}^+ \mapsto \mathcal{X}$, where w is a random process in the probability space Ω. That is, $x(t+\tau) = F(x(t), w, \tau)$ when the control law f is applied.

The functions $\phi : \mathcal{X} \mapsto \{\mathsf{true}, \mathsf{false}\}$ and $\psi : \mathcal{X} \mapsto \{\mathsf{true}, \mathsf{false}\}$ correspond to *precondition* and *postcondition*, respectively. The precondition $\phi(x)$ equals true if this EA is admissible at x, and false otherwise. The postcondition $\psi(x)$ indicates the nominal result that the EA is designed to achieve, with $\psi(x) = \mathsf{true}$ when x is in the nominal range of the terminal state of this EA. The number $\hat{p}$ is a level of probability, and the number Θ is a finite duration in time. Taken together, the execution of the control law f with initial state in $\phi^{-1}(\mathsf{true})$ satisfies [3] :

(1) The system will not go to ruin under f, where "ruin" is defined as a domain $\mathcal{R}$, a subset of $\mathcal{X}$ that is disjoint from both $\phi^{-1}(\mathsf{true})$ and $\psi^{-1}(\mathsf{true})$, and such that when x enters $\mathcal{R}$, it becomes infeasible for any control law to drive x out of $\mathcal{R}$.
(2) The system state x will enter $\psi^{-1}(\mathsf{true})$ from $\phi^{-1}(\mathsf{true})$ in less than Θ units of time with a probability greater than $\hat{p}$. More precisely, for all t,

$$\Pr\begin{pmatrix} \exists\delta \in (0,\Theta), \\ \phi(F(x(t),w,\tau)) = \mathsf{true} \text{ for } 0 \le \tau < \delta, \\ \text{and } \psi(F(x(t),w,\delta)) = \mathsf{true} \end{pmatrix} > \hat{p}. \qquad (2)$$

[3] Here $\phi^{-1}(\mathsf{true})$ and $\phi^{-1}(\mathsf{false})$ denote the preimage of true and false, respectively, although ϕ^{-1} as a function might not exist. The same is true for ψ^{-1}.

In plain words, when admissible, an EA is safe and has a chance to succeed in some given time. We will call $\hat{p}$ the *minimum success probability*, Θ the *nominal duration*, and $\phi^{-1}(\mathsf{true})$ the *admissible domain* of this EA. When we say an EA is taken, applied, or active, it should be understood that the associated control law $u = f(x)$ is applied.

The ideas of preconditions and postconditions in the deterministic context are standard in Artificial Intelligence. But they have not been widely adopted in the control of mechanical systems such as autonomous robots. This is to a large extent due to the uncertainties associated with such systems. The inclusion of a success probability helps us connect AI-type planning with feedback control under uncertainties. This point will be explored in Section 4 and 5.

3.2 Order-Based Policies and Their Coding Complexity

We next construct the *order-based policies* using the EAs and their preconditions.

Suppose the mission is to drive the system such that a boolean-valued function $\phi^*(x) : \mathcal{X} \mapsto \{\mathsf{true}, \mathsf{false}\}$ becomes true, after which the mission ends. Suppose we have an ordered set of EAs, $\mathcal{A} = \{EA_1, EA_2, \ldots, EA_L\}$, and for all $x \in \mathcal{X}$ and $\phi^*(x) = \mathsf{false}$, there is an EA, $EA_i \in \mathcal{A}$ such that $\phi_i(x) = \mathsf{true}$. That is, there is always an admissible EA before the mission is accomplished. A feedback policy can be constructed via the algorithm shown in Figure 2, which we will call Algorithm 1.

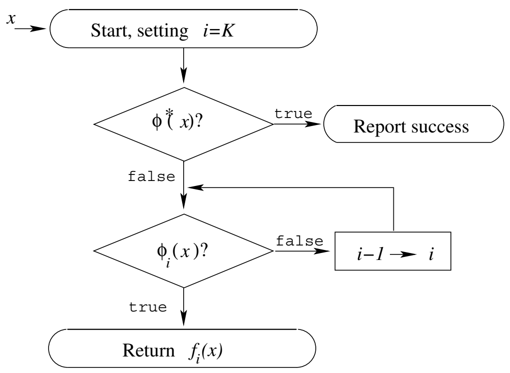

Fig. 2. Flowchart of Algorithm 1.

In effect, the control policy is

$$\begin{aligned} &u = f_i(x) \quad \text{if} \quad \phi_i(x) = \mathsf{true}, \text{ and} \\ &\phi_{i+1}(x) = \ldots = \phi_L(x) = \phi^*(x) = \mathsf{false}. \end{aligned} \qquad (3)$$

For this policy to be well-defined, we assume for all $x \in \mathcal{X} - \mathcal{R}$ and $\phi^*(x) \neq \mathsf{true}$, there is at least one index $i = 1, \ldots, L$ such that $\phi_i(x) = \mathsf{true}$.

Here, the ordering of the EAs can be understood as a notion of priority, with a higher index associated with a higher priority. In other words, this policy picks an EA to apply if this EA is admissible and no EA with higher priority is admissible. Note that each order-based policy candidate corresponds to a permutation of $\mathcal{A}$ (although not necessarily a unique permutation).

Certainly, given a fixed set of EAs, order-based policies do not cover all possible feedback policies. We will not discuss this in detail here due to the space limitation. However, one can expand the range of achievable behaviors by splitting one EA into two or more and inserting them to different places in the ordered EA set. This translates to a way of managing complexity.

3.3 Managing Complexity/Richness

First, as shown in Section 2.2, given a set of EAs, $\mathcal{A} = \{EA_1, EA_2, \ldots, EA_L\}$, the coding complexity under the conventional hierarchical approach is

$$\mathbb{CC}(\mathcal{B}_1) = 2^{N_1} \cdot \log_2 L,$$

where N_1 can be easily over one hundred. Now, if we focus on the order-based policies, the range of candidates, denoted by $\mathcal{B}_2$, has cardinality no greater than the total number of permutations of $\mathcal{A}$. The coding complexity is then

$$\mathbb{CC}(\mathcal{B}_2) \le \log_2 L! < L \cdot \log_2 L.$$

To compare $\mathbb{CC}(\mathcal{B}_1)$ and $\mathbb{CC}(\mathcal{B}_2)$, note that 2^{N_1} is the cardinality of the discretized state space, which is usually extremely large (10^{39} in the example in Section 2), whereas L is usually much smaller, likely in the tens. Thus, focusing on order-based policies gives us a very substantial reduction of complexity. Of course, this would not be meaningful if we have also thrown away what the system is designed to be capable of. We will address this concern in later sections with very reassuring evidences.

Second, it is possible to manage complexity (or richness) within order-based policies by "splitting" the EAs. That is, from one EA, $(f, \phi, \psi, \hat{p}, \Theta)$, one may derive two EAs without redesigning the control law: $(f, \phi', \psi', \hat{p}', \Theta')$ and $(f, \phi'', \psi'', \hat{p}'', \Theta'')$. Developing systematic schemes for doing this is a promising subject for future research. For now, first note that any control policy in $\mathcal{A}^{\mathcal{X}_1}$ can be achieved as an order-based policy by splitting the initial EAs in the afore mention way. In the extreme case, the original domain of each EA would be chopped into singletons. (Of course, this is not what we intend to do.) More interestingly, note that the minimum increment of complexity due to splitting an EA is $\log_2(L+1)! - \log_2 L! = \log_2(L+1)$, which is on the magnitude of one-Lth of the original complexity. In comparison, under the conventional hierarchical approach, the richness of the control policy is typically enhanced by augmenting the state space (increase the precision of existing state variables or adding new internal states). By adding a single digit to the state word, the coding complexity increases to $2^{N_1+1} \log_2 L$, which is a two-fold increase. Thus, working with order-based framework provides a much better way for fine-tuning the richness of the system behavior.

4. TIME-TO-ACCOMPLISH ANALYSIS

To analyze the time-to-accomplish in the presence of uncertainties, probabilistic information of the robot and its environment is needed. But complete knowledge of such information would be unrealistic to assume. Here, we only use the minimum success probabilities of the EAs. This led us to consider processes with uncertain transition probabilities.

Let us first cite a result from Li et al. (2007a). Consider a stochastic process with $L+1$ possible states, where the state $L+1$ is absorbing, i.e., when the system state reaches $L+1$, it will remain at that value perpetually without control. Let $\pi(k) = (\pi_1(k), \ldots, \pi_L(k))^T$, where $\pi_i(k)$, $i = 1, \ldots, L$ is the probability of reaching state i at time k, and let $\pi^*(k)$ be the probability of reaching the absorbing state by time k. Let $P(k)$ be the transition probability matrix with the row and column corresponding to the absorbing state removed. $P(k)$ is thus a substochastic matrix. This shall be understood when we mention probability transition matrices in what follows. Thus

$$\pi(k+1) = P(k)\pi(k), \quad \pi^*(k) = 1 - \mathbf{1}^T\pi(k). \qquad (4)$$

Note that the transition probability matrix is possibly time-varying.

Definition 2. (Progressivity). Consider two $L \times L$ substochastic matrices P_1 and P_2. We say that P_2 is *strictly more progressive* than P_1, denote by $P_2 \succ P_1$, if $q^T(P_2 - P_1) < 0$ (elementwise) for any vector $q = (q_1, q_2, \ldots, q_L)^T$ such that $q_1 > q_2 > \ldots > q_L > 0$.

Definition 3. (Sortedness). For a constant probability transition matrix P, let $\rho = (\rho_1, \rho_2, \ldots, \rho_L)^T$ be the nonnegative left eigenvector associated with the maximal eigenvalue r, i.e., $\rho^T P = r\rho^T$. We say that P is *sorted* if $\rho_1 > \rho_2 > \ldots > \rho_L$.

Lemma 1. [Li et al. (2007a)] Let $\hat{P}$ be a constant matrix that is irreducible, sorted, and has a maximal eigenvalue $r < 1$. If $P(k) \succ \hat{P}$ for all $k \ge 0$, then $(1 - \hat{\pi}^*(k))r^{-k} \to 0$ for $t \to \infty$.

We are now ready to analyze the time-to-accomplish of the order-based policies.

Theorem 1. Consider an ordered set of L EAs executed via Algorithm 1 (which constitute an order-based policy). Let the mission be to render a boolean-valued function $\phi^*(x)$ to become true. If for each index $i = 1, \ldots, L$, there is a $i < i' \le L+1$ such that $\psi_i(x) = \texttt{true}$ implies $\phi_{i'}(x) = \texttt{true}$ (or $\phi^*(x) = \texttt{true}$ if $i' = L+1$), then this mission is accomplished w.p.1 for any initial condition $x_0 \in \mathcal{X} - \mathcal{R}$. Further, the probability of mission-not-accomplished by time t, as $t \to \infty$ is less than

$$r^{t/\Theta_M},$$

where $0 < r < 1$ is the maximal eigenvalue of the matrix

$$\hat{P} = \begin{pmatrix} 1-\hat{p}_1 & 1-\hat{p}_2 & 1-\hat{p}_3 & \cdots & 1-\hat{p}_k \\ \hat{p}_1 & 0 & 0 & \cdots & 0 \\ 0 & \hat{p}_2 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & \cdots & \hat{p}_{k-1} & 0 \end{pmatrix},$$

and $\Theta_M = \max\{\Theta_1, \Theta_2, \ldots, \Theta_L\}$. (Recall that the $\hat{p}_i$'s and Θ_i's are the minimum success probabilities and nominal durations of the EAs.)

Proof:(Only a sketch due to page limit.) The execution of a policy based on a set of L EAs generates a symbolic sequence $s(k) : \mathbb{R}^+ \mapsto \{1, 2, \ldots, L, L+1\}$, which records the EAs actually taken to accomplish the mission. When the mission starts, $s(0)$ is generated to record the first EA applied. Each next symbol is generated when the policy switches to a different EA, or the EA currently applied reaches its nominal duration. Mission accomplishment is recorded as $s(k) = L+1$, and is an absorbing state of the sequence. Let t_k be the time when the kth symbol is generated. Then,

$$\frac{t_k}{k} \le \Theta_M. \qquad (5)$$

The probability transition of $s(k)$ can be written in the form of (4):

$$\phi(k+1) = P(k, x_0)\phi(k), \quad \psi(k) = 1 - \mathbf{1}^T\phi(k),$$

where $\phi(k)$ is a L-dimensional vector recording the probability distribution of $s(k)$ over 1 through L, and ψ is the probability of the mission being accomplished by time t_k. One can show that the EAs having minimum success probabilities translates to that $P(k)$ is strictly more progressive than $\hat{P}$ for all k.

The matrix $\hat{P}$ is also sorted. To see this, let r be the maximal eigenvalue of $\hat{P}$. From the property of irreducible matrices, r is greater than the maximal eigenvalue of any principal submatrix of $\hat{P}$ (see Minc (1988)). Thus $r > 1 - \hat{p}_1$. One may verify that the eigenvector of $\hat{P}$ associated with r is $q = (q_1, q_2, \ldots, q_k)^T$, with

$$q_1 = 1,$$
$$q_i = 1 - (1-r)\frac{r^{i-2} + r^{i-3}p_1 + \ldots + p_1 p_2 \cdots p_{i-2}}{p_1 p_2 \cdots p_{i-1}},$$
$$i = 2, \ldots, k.$$

Using $1 - p_1 < r < 1$ and with some tedious calculations, it can be shown that $q_1, q_2, \ldots, q_k$ is indeed a decreasing sequence. So, the process $s(k)$ converges to $L+1$ in probability at a rate greater than that of r^k. Almost sure convergence to mission accomplishment follows based on the well-known Borel-Cantelli Lemma (Feller (1968)).

■

Since $P(k, x_0)$ may approach $\hat{P}$ under the hypothesis of this theorem, the bound is tight if the nominal durations of the EAs are uniform.

5. CONNECTION TO PLANNING

The simplest notion of a plan is a sequence of actions that each paves the way for the subsequent ones and finally leads to a goal. Formally, consider again a mission represented by the boolean-valued function $\phi^*(x)$. By a *plan* for this mission, we mean a finite sequence of EAs, $EA_1, EA_2, \ldots, EA_L$, such that

(1) Given x, $\psi_L(x) = \mathsf{true}$ implies $\phi^*(x) = \mathsf{true}$.
(2) Given x, $\psi_i(x) = \mathsf{true}$ implies $\phi_{i+1} = \mathsf{true}$, $i = 1, \ldots, L-1$.

Assume for all $x \in \mathcal{X} - \mathcal{R}$ such that $\phi^*(x) = \mathsf{false}$, there is an index $i \in \{1, 2, \ldots, L\}$ such that $\phi_i(x) = \mathsf{true}$. This plan will achieve the mission w.p.1 when executed through Algorithm 1, according to Theorem 1.

However, a single plan like this may not be enough. Very often, we humans keep a Plan A and a Plan B, or even a Plan C and a Plan D, etc. The hypothesis of Theorem 1 leaves enough room for integrating multiple plans. Namely, if a collection of plans covers all possible state values (each individual one does not have to), and if we merge the plans while maintaining the relative order of the EAs within the individual plans, then the merged sequence satisfies the hypothesis of Theorem 1. See Figure 3. Note that this allows more than one way of merging multiple plans. To further determine the optimal merged sequence, we will need to gather more information of the transition probabilities. The concept of sortedness will also be useful for optimizing the merged sequence. This idea will be explored in our future work.

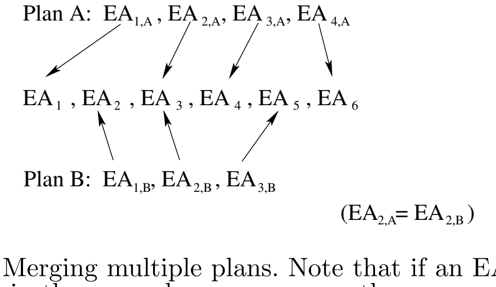
Plan A: EA1,A , EA2,A, EA3,A, EA4,A
EA1 , EA2 , EA3 , EA4 , EA5 , EA6
Plan B: EA1,B, EA2,B , EA3,B
(EA2,A= EA2,B)

Fig. 3. Merging multiple plans. Note that if an EA shows up in the merged sequence more than once, only the highest indexed appearance needs to be kept.

6. THE TWO-ON-TWO ROBOT SOCCER EXPERIMENT

The system used for this experiment was contributed by the former Cornell University RoboCup Team, who had won four Championships in the International RoboCup Competition, F180 (small size) League. The legacy control code (written in C++) is composed of five layers:

- *Local control.* A loop directly connected to the wheel encoders, the gyroscope, and the motors, etc.
- *Trajectory generation.* Moving from one point to another with obstacle avoidance.
- *Skills.* Scripts for shooting the goal, making passes, maneuvering with the ball, etc.
- *Roles.* Similar to the skills, but somewhat more complicated, sometime calling upon multiple skills.
- *Role assignment.* This is the highest layer, which assigns roles to the robots in play. This was done by a very large piece of code (over 1000 lines).

The legacy code has a total of more than 30 active skills and roles, which are not equipped with preconditions and postconditions. In general, the control laws at all layers directly reference the position and velocity of the ball and of every robot in the field. See Sherback et al. (2006) for more information on the Cornell RoboCup system.

Our new order-based control code has only three layers: *local control, elementary actions (EAs),* and *action plan.* The local control was the same as the legacy code, but the EAs and the action plan are almost all new. If we try to link them to the structure of the legacy code, the EA-layer combines trajectory generation, skills, and roles, and the action plan replaces role assignment. The new code is much more transparent, while at the same time, it competed at par against the legacy code in a 2-on-2 game.[4]

6.1 EAs for robot soccer

The EAs of the new program have roughly the same level of complexity as the skills in the legacy code. Our progress in designing the EAs has been reported in Li and D'Andrea (2006). Note that these EAs are designed analytically, which confirms our view of the sources of complexity (Section 2.2). The EAs used in our new control code are:

- Defense
- Wander around a certain point in the field
- Acquire ball to maneuver
- Acquire ball for passing to a teammate
- Acquire ball for shooting the goal
- Maneuver the ball to a certain area
- Pass the ball to a teammate
- Shoot the goal
- Avoid collision

Recall that the reduction of complexity with the order-based policies rely on the assumption that the number of bits for encoding the EAs is much less than the number of bits for encoding the state of the robot's world. Here, we only use 9 EAs, which takes 4 bits to encode. The state in this case includes the position, orientation, and velocity of the robots (4 of them), and the position and velocity of the ball, plus various internal states. At least 112 bits are needed if the positions, orientations, and velocities are encoded with 8 bits each, not counting the internal states.

Somewhat more programming is needed to write the preconditions though. To give a glimpse of what the preconditions look like, the precondition of acquire-ball-for-shooting-the-goal (function $\phi(x)$ of that EA) returns true for one robot when:

(1) the ball is not out of bounds,
(2) no teammate is in possession of the ball,
(3) "me" is in the best position to get the ball,

[4] In fact, the new program is also capable of playing an exciting 5-on-5 game, as was the format of the International RoboCup Competition. But due to the cost of fabricating and maintaining additional hardware, we have run the 5-on-5 game only in simulations.

(4) no player on the opposing team is in possession of the ball, and
(5) no player on either team is blocking the path for "me" to shoot the goal.

Programming the preconditions was not very difficult since it simply spells out what the control laws have already assumed.

6.2 *Order-Based policy for robot soccer*

For each player, our new policy arranges the EAs in the following order:

(1) Wander around a designated point in the field
(2) Acquire ball to maneuver
(3) Maneuver the ball to a designated area
(4) Acquire ball for passing to a teammate
(5) Pass the ball to a teammate
(6) Defense
(7) Avoid collision
(8) Acquire ball for shooting the goal
(9) Shoot the goal

Although it appears simple, this sequence can be interpreted as a combination of multiple plans.

Plan A: Wander around $\longrightarrow$ Acquire ball to maneuver $\longrightarrow$ Maneuver to an area that has advantage for either making a pass or shooting the goal $\longrightarrow$ Acquire ball for shooting the goal $\longrightarrow$ Shoot the goal.

Plan B: Acquire ball for passing $\longrightarrow$ Pass the ball to teammate $\longrightarrow$ Teammate acquires ball for shooting $\longrightarrow$ Teammate shoots the goal.

Plan C: Defense (taking the ball away from the opponent) $\longrightarrow$ Acquire ball for shooting the goal $\longrightarrow$ Shoot the goal.

Plan D: Same as Plan A, but add Avoid collision after maneuvering.

The interpretation in terms of plans is not unique, and indeed, the actual sequence of actions may not follow any of these plans. The ordering of the EAs can also be modified to generate different robot behaviors in the game. For example, we can lower the indices of "defense" and "avoid collision" for a more aggressive game play.

6.3 *Experiment result and discussion*

We have run the new control program against the legacy program with actual robots in the 2-on-2 game several times, and the matches seemed close. See Li et al. (2007b) for a 5-minute clip in which the new program won by 5 to 4. The new program did not always win, though. We combined the scores of several pieces of experiment[5] with a total time of 18 minutes, and the legacy code won by 20 to 18. But in any case, it was clear that they were competing at the same level. Considering that the legacy code belongs to one of the best teams in robot soccer, this is an intriguing result.

So far, we have not applied the analysis of sortedness to this soccer game because we have not implemented any statistical measurement of the transition probabilities. But this certainly can be done using various techniques (e.g., particle filtering). We leave it to future work. We also have not implemented the management of complexity in this experiment. As explained earlier in this paper, the management of complexity can be done by splitting the EAs. This is analogous to the human intellectual activity of discovering concepts, which have the function of distinguishing between situations that were previously treated as one. This is also a rich and promising topic for future research.

[5] The experiment was interrupted at times due to the need for hardware maintenance. The 5-minute clip is the longest uninterrupted one among these pieces.

7. CONCLUSION

This paper formulated a scheme of layered autonomous robot control, which has the novel feature that the upper-layer policy is an ordering of the lower-layer modules, not a function of the state of the robot's "world". This approach was motivated by the need for managing complexity. We further discussed performance analysis within the proposed scheme given partial (almost minimum) probabilistic information. Experiment result based on a 2-on-2 soccer game was also provided, in which our new code competed at par against the legacy code of the former championship team of the International RoboCup Competition.

REFERENCES

S. Andersson and D. Hristu-Varsakelis. Stochastic language-based motion control. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3313–18, 2003.

R. W. Brockett. Formal languages for motion description and map making. In J. Baillieul, D.P. Martin, R. W. Brockett, B.R. Donald, and R.M. Murray, editors, *Robotics*, pages 181–93. American Mathematical Society, 1990.

R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

J. Bryson. The behavior-oriented design of modular agent intelligence. In J. P. Muller, editor, *Proceedings of Agent Technology and Software Engineering*, 2002.

E. de Sevin and D. Thalmann. A motivational model of action selection for virtual humans. In *Proceedings of Computer Graphics International 2005 (CG1'05)*, Stony Brook, NY, 2005.

M. Egerstedt. Motion description language for multimodal control in robotics. In A. Bicchi, H. I. Christensen, and D. Prattichizzo, editors, *Control Problems in Robotics*, pages 75–89. Springer-Verlag, Berlin Heidelberg, 2003.

W. Feller. *An Introduction to Probability Theory and Its Application.* John Wiley & Sons, 1968.

D. Isla, R. Burke, M. Downie, and B. Blumberg. A layered brain architecture for synthetic creatures. In *Proceedings of IJCAI*, Seattle, WA, August 2001.

J. F. Lehman, J. Laird, and P. Rosenbloom. A gentle introduction to soar: 2006 update. Online, 2006.

K. Li and R. D'Andrea. Trajectory design of autonomous vehicles based on motion primitives and heuristic cost-to-go. In *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 2006.

K. Li, S.-C. Kang, and I. Paschalidis. Some results on the analysis of stochastic processes with uncertain transition probabilities and robust optimal control. In *Proceedings of the 45th Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2007a. Also submitted to IEEE Transactions on Automatic Control.

K. Li, O. Purwin, and R. D'Andrea. Two-on-two robot soccer game. On-line: http://hdl.handle.net/1813/8238, 2007b.

V. Manikonda, J. Hendler, and P. S. Krishnaprasad. Languages, behaviors, hybrid architectures and motion control. In J. B. Baillieul and J. C. Willems, editors, *Mathematical Control Theory*, pages 199–226, 1998.

H. Minc. *Nonnegative Matrices.* Wiley-Interscience, 1988.

M. Sherback, O. Purwin, and R. D'Andrea. *Real-time motion planning and control in the 2005 Cornell RoboCup system*, pages 245–264. Springer Verlag, 2006.

I. H. Suh, M. J. Kim, S. Lee, and B. J. Yi. A novel dynamic priority-based action selection mechanism integrating a reinforcement learning. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, New Orleans, LA, 2004.